HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 10011596-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Christophe de Dinechin et al.

Confirmation No.: 5117

Application No.: 09/873,875

Examiner: Lillian Vo

Filing Date: June 4, 2001

Group Art Unit: 2195

Title: CONTEXT-CORRUPTING CONTEXT SWITCHING

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on 6-20-2005.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

( )	one month	$120.00
( )	two months	$450.00
( )	three months	$1020.00
( )	four months	$1590.00

( ) The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of **$500.00**. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

( ) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit:_____

OR

(X) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (703) 872-9306 on 6/20/2005

Number of pages: 17

Typed Name: Hugh Gortler

Signature: Hugh Gortler

Respectfully submitted,

Christophe de Dinechin et al.

By Hugh Gortler

Attorney/Agent for Applicant(s)
Reg. No. 33,890

Date: 6/20/2005

Telephone No.: (949) 454-0898

Rev 12/04 (Apbrief)

Patent
Docket No. 10011596-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

APPEAL NO. _____

In re Application of:
Christophe de Dinechin et al.

Serial No. 09/873,875
Filed: June 4, 2001

Confirmation No. 5117
Examiner: Lillian Vo
Art Unit: 2195

For: **CONTEXT-CORRUPTING CONTEXT SWITCHING**

APPEAL BRIEF

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents Washington, D.C. 20231 on June 20, 2005.

\Hugh Gortler #33,890\
HUGH P. GORTLER, Reg. No. 33,890

Hugh P. Gortler, Esq.

Hewlett-Packard Company
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

(949) 454-0898

INDEX

		Page
1.	REAL PARTY IN INTEREST	1
2.	RELATED APPEALS AND INTERFERENCES	1
3.	STATUS OF CLAIMS	1
4.	STATUS OF AMENDMENTS	1
5.	SUMMARY OF CLAIMED SUJECT MATTER	1
6.	GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL	5
7.	ARGUMENTS	5
	a. Rejection of claims 1, 12 and 22 under 35 U.S.C 103(a) over Bugnion et al U.S. Patent No. 6,496,847	5
	b. Rejection of claim 11 under 35 U.S.C 103(a) over Bugnion et al U.S. Patent No. 6,496,847	7
	c Rejection of claims 1-11 and 22-26 under 35 U.S.C 101	8
8.	CLAIMS APPENDIX	10
9.	EVIDENCE APPENDIX	None
10.	RELATED PROCEEDINGS APPENDIX	None

1. REAL PARTY IN INTEREST

The real party in interest is the assignee, Hewlett-Packard Development Company.

2. RELATED APPEALS AND INTERFERENCES

No appeals or interferences are known to have a bearing on the Board's decision in the pending appeal.

3. STATUS OF CLAIMS

Claims 1-26 are pending.

Claims 1-26 are rejected.

The rejections of claims 1-26 are being appealed.

4. STATUS OF AMENDMENTS

No amendment was filed subsequent to final rejection.

5. SUMMARY OF CLAIMED SUBJECT MATTER

An operating system (OS) can be run as application on a computer using a virtual machine. The OS that is run as the application is referred to as the "guest OS," and the underlying OS is referred to as the "host OS." For example, a Windows-based OS can be run as an application on top of a host Linux OS. Running the Windows-based OS as an application allows programs written for a Windows environment to be run on top of the host Linux OS.

The guest OS usually has a different "context" than the host OS. A context embodies the accessible state of the computer's central processing unit (CPU).

The context includes in particular the values of CPU registers. It does not include resources that the CPU would not allow currently executing code to access.

The virtual machine can allow the guest OS and the host OS to run concurrently by properly restoring the context of the guest OS whenever control is transferred from the host OS to the guest OS, and by similarly restoring the context of the host OS when control is transferred from the guest OS to the host OS. This process is called "context switching."

Certain CPU architectures, such as IA-64, have prohibitions against, and difficulties with, completely saving and restoring the entire context of a guest OS or a host OS. Specifically, certain registers containing context might be corrupted during context switching. If the entire context cannot be saved, the guest OS or host OS might behave incorrectly and crash.

Paragraph 17 of the application provides a simple example. Context of a host OS is contained in registers X, I, J and K. During context switching, however, register X is used to store an address that indicates where the context will be saved. Thus, the register X is overwritten during context switching, whereby the context of register X is lost before it can be saved.

This problem is overcome by the method of claim 1, the method of claim 11, the apparatus of claim 12, and the software of claim 22.

Independent claim 1

Claim 1 recites a method of switching context on a processor (element 110 in Figure 1). Referring to Figure 2 and paragraphs 20-21 of the application, the method comprises saving the context under software control using an

inconsequential register (312); and preventing the processor from changing the context while the context is being saved (310).

The inconsequential register can be used to store context, for example, by storing an address that indicates where the context will be saved. This example is described in paragraph 23 of the application.

According to paragraph 21 of the application, an inconsequential register is a register that is not used by the host OS at a predetermined interruption point (PIP). A point can be predetermined if the context is saved under software control (which is asynchronous) instead of hardware control (which is asynchronous). According to paragraph 22 of the application, since the context of a host OS is saved at the PIP, it is known which registers the host OS uses and which registers the host OS does not use. Therefore, the inconsequential register(s) at the PIP can be identified. The inconsequential registers can be corrupted by a virtual machine application without affecting the host OS. In particular, the context switching process can use the inconsequential registers as a temporary storage in lieu of the privileged registers.

Thus, writing to ***any*** hardware register will not solve the problem posed in the application. If ***any*** register is used, the entire context might not be saved, and the guest OS or host OS might behave incorrectly and crash

Independent claim 12

Reference is made to Figures 1-2. Claim 12 recites apparatus comprising a processor (110) including a plurality of registers (110) and a virtual machine application (212). The virtual machine application (212) commands the processor (110) to switch context by saving the context under software control

using an inconsequential register of the processor as temporary storage (312). The virtual machine application also prevents the processor from changing the context while the context is being saved (310).

Independent claim 22

Reference is made to Figures 1-2. Claim 22 recites software (212) comprising instructions for commanding a processor to switch context by saving the context under software control using an inconsequential register of the processor as temporary storage (312); and preventing the processor from changing the context while the context is being saved (310).

Independent claim 11

Reference is made to Figures 1-2 and paragraphs 20-22 of the application. Claim 11 recites a method of switching context between a host OS (210) and a virtual machine (212) on a processor (110). The processor (110) has privileged registers and access to other memory (112). Referring additionally to Figure 3 and paragraph 29, the method comprises giving the virtual machine access to the privileged registers (point A in Figure 3); using at least one privileged register as temporary storage to save the context in the other memory at a predetermined interruption point (between points A and B in Figure 3); and preventing the processor from changing the context while the context is being saved (310). The virtual machine application controls the context switch.

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

a. Claims 1, 12 and 22 are rejected under 35 USC §103(a) as being unpatentable over Bugnion et al. U.S. Patent No. 6,496,847.

b. Claim 11 is rejected under 35 USC §103(a) as being unpatentable over Bugnion et al. U.S. Patent No. 6,496,847.

c. Claims 1-11 and 22-26 are rejected under 35 USC §101 as being directed to non-statutory subject matter.

7. ARGUMENTS

REJECTION OF CLAIMS 1, 12 AND 22 UNDER 35 U.S.C §103(a) OVER BUGION ET AL. U.S. PATENT NO. 6,496,847

The office action acknowledges that Bugnion et al. do not teach the use inconsequential registers for context switching. However, the office action contends that such use is obvious because Bugnion et al. state that ***any*** hardware register can be used to save context.[1]

The use of ***any*** register will not solve the problem posed in the application. Consider once again the example in paragraph 17 of the application. If Bugnion's teachings are followed, and ***any*** register is used, the context of register X could become lost before it can be saved. If context is lost, the guest OS or host OS might behave incorrectly and crash.

---

[1] At col. 11, lines 39-41, Bugnion et al. state "Any available memory space may be used to save this information, and actual storage and retrieval may be accomplished using any known technique."

The office action's legal analysis is flawed. Bugnion et al. do not teach or suggest the use of inconsequential registers; therefore, the examiner substitutes unsubstantiated opinion for evidence of suggestion in the prior art. However, an examiner's unsubstantiated allegations with respect to knowledge in the prior art does not provide evidence of suggestion, particularly in light of a challenge. See In re Ahlert, 424 F2d.,1088, 1091-92 165 USPQ 418, 420-421 (CCPA1970). The examiner's unsubstantiated opinion was challenged in the previous response, and it is being challenged now. Specifically, the evidence made of record does not teach or suggest context switching that uses registers that have been identified as inconsequential.

If inconsequential registers are used under software control, while the processor is prevented from changing the context, the entire context is saved and a system crash is prevented. Thus, the method of claim 1, the apparatus of claim 12, and the software of claim 22 clearly recite features that advance the art over the teachings of Bugnion et al.

Bugnion et al. do not identify the context switching problem that is identified in the application. Bugnion et al. do not teach or suggest a solution to this problem, let alone the solution recited in claims 1, 12 and 22. Therefore, the '103 rejections of independent claims 1, 12 and 22 and their dependent claims over Bugnion et al. should be withdrawn.

None of the other documents made of record teach or suggest the use of inconsequential registers during context switching. Therefore, claims 1-10 and 12-26 should be allowed over the documents made of record.

REJECTION OF CLAIM 11 UNDER 35 U.S.C §103(a) OVER BUGNION ET AL. U.S. PATENT NO. 6,496,847

Bugnion et al. don't teach or suggest saving context at a predetermined interruption point. Moreover, Bugnion et al. do not teach or suggest the use of privileged registers as temporary storage to save context in other memory (e.g., RAM).

The office action cites a passage at col. 4, lines 52-61. The passage discloses that host OS context is saved in a driver, and then context switching is performed in the driver. The passage also states that virtual machine monitor context is saved in a virtual machine monitor, and then context switching is done in the virtual machine monitor. However, the passage does not indicate how the context is saved.

The office action cites a passage at col. 4, lines 46-48. The passage states that a computer uses a set of privileged instructions. However, the passage says nothing about privileged registers being used as temporary storage to save context.

The office action cites a passage at col. 11, lines 13-15. The passage states that code is executed when an interrupt occurs. The passage does not indicate whether context is switched at a synchronous interrupt (whereby a predetermined interrupt point can be identified) or an asynchronous interrupt.

The office action cites a passage at col. 11, lines 30-52. This passage gives an overview of Bugnion et al.'s context switch. The passage indicates that any available memory may be used to save context. The passage does not

indicate whether context is saved at a predetermined interruption point, or whether privileged registers are used as temporary storage to save context.

Thus, the office action has not established prima facie obviousness of claim 11. Therefore, claim 11 should be allowed over Bugnion et al. alone.

REJECTION OF CLAIMS 1-11 and 22-26 UNDER 35 U.S.C §101

The office action contends that the methods of claims 1-11 and 23-26 can be carried out as a mental process in conjunction with pen and paper. Claim 1 recites storing data in hardware registers. Claim 1 also recites preventing a processor from changing contexts. The office action doesn't explain how pen and paper can be used to store processor context in a hardware register, or how pen and paper can be used to prevent a processor from changing context while the context is being saved. Because claim 1 clearly recites actions that do not read on a mental process, the '101 rejections of claims 1-11 and 23-26 should be withdrawn.

The office action contends that claim 22 lacks utility and is not tangibly embodied in a manner to be executable. This contention is puzzling, since claim 22 recites "software comprising instructions for commanding a processor to switch context." Context switching is a useful operation; therefore, the software of claim 22 has utility. The software can command a processor; therefore, it has a tangible embodiment that can be executed. Withdrawal of the '101 rejection of claim 22 is respectfully requested.

For the reasons above, all pending claims should be allowed over Bugion alone. Moreover, all claims recite statutory subject matter. The Honorable Board of Patent Appeals and Interferences is respectfully requested to reverse the '101 and '103 rejections.

Respectfully submitted,

\Hugh Gortler #33,890\
Hugh P. Gortler, Esq.
Registration No. 33, 890

Hewlett-Packard Company
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

(949) 454-0898

Date: June 20, 2005

8. CLAIMS APPENDIX

1. (Previously presented) A method of switching context on a processor, the method comprising:

saving the context under software control using an inconsequential register; and

preventing the processor from changing the context while the context is being saved.

2. (Original) The method of claim 1, wherein the inconsequential register is used as a temporary storage in lieu of a privileged register.

3. (Original) The method of claim 1, wherein the context is saved at a predetermined interruption point.

4. (Original) The method of claim 1, wherein the context is switched between a host operating system and a virtual machine application, the virtual machine application controlling the context switch.

5. (Original) The method of claim 4, wherein the inconsequential register is used to pass information to the virtual machine application.

6. (Original) The method of claim 1, wherein the context is switched using an IA-64 processor.

7. (Previously presented) The method of claim 6, wherein the inconsequential register is a caller-save register.

8. (Previously presented) The method of claim 6, wherein the inconsequential register is a branch register.

9. (Original) The method of claim 1, further comprising restoring the context using the inconsequential register as temporary storage.

10. (Original) The method of claim 9, wherein the context is restored by using a branch register to perform an indirect branch.

11. (Previously presented) A method of switching context between a host OS and a virtual machine on a processor, the processor having privileged registers, the processor having access to other memory, the method comprising:

giving the virtual machine access to the privileged registers;

using at least one privileged register as temporary storage to save the context in the other memory at a predetermined interruption point; and

preventing the processor from changing the context while the context is being saved;

the virtual machine application controlling the context switch.

12. (Original) Apparatus comprising:

a processor including a plurality of registers; and

a virtual machine application for commanding the processor to switch context by saving the context under software control using an inconsequential register of the processor as temporary storage; and preventing the processor from changing the context while the context is being saved.

13. (Original) The apparatus of claim 12, wherein the inconsequential register is used as a temporary storage in lieu of a privileged register.

14. (Original) The apparatus of claim 12, wherein the context is saved at a predetermined interruption point.

15. (Original) The apparatus of claim 12, further comprising a host OS; wherein the context is switched between the host OS and the virtual machine application; and wherein the virtual machine application controls the context switch.

16. (Original) The apparatus of claim 15, wherein the inconsequential register is used to pass information to the virtual machine application.

17. (Original) The apparatus of claim 12, wherein the processor is an IA-64 processor.

18. (Previously presented) The apparatus of claim 17, wherein the inconsequential register is a caller-save register.

19. (Previously presented) The apparatus of claim 17, wherein the inconsequential register is a branch register.

20. (Previously presented) The apparatus of claim 12, wherein the virtual application further commands the processor to restore context using the inconsequential register as temporary storage.

21. (Original) The apparatus of claim 20, wherein the context is restored by using a branch register to perform an indirect branch.

22. (Original) Software comprising instructions for commanding a processor to switch context by saving the context under software control using an inconsequential register of the processor as temporary storage; and preventing the processor from changing the context while the context is being saved.

23. (Previously presented) The method of claim 1, wherein content of the inconsequential register is corrupted during the context switch.

24. (Previously presented) The method of claim 1, wherein using the inconsequential register includes storing an address in the inconsequential register, the address indicating a memory location at which the context will be saved.

25. (Previously presented) The method of claim 1, wherein the inconsequential register does not store context at a predetermined interruption point.

26. (Previously presented) The method of claim 1, wherein the context is stored in memory other than the inconsequential register.